QuickTime APIs Update

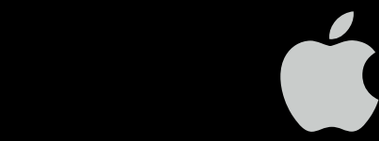
Session 708

QuickTime APIs Update

Kevin Calhoun
Software Engineer
kcalhoun@apple.com

Introduction

- QuickTime offers:
  - Playback
  - Capture
  - Creation and Editing
  - Delivery
- QuickTime's platform citizenship
- What is new for developers in QuickTime 6.x



What You Will Learn

- What QuickTime can do in your application
- Media facilities for the modern OS
- What has changed and how QuickTime is changing with it

Technology Framework

- QuickTime for…
  - Carbon
  - Cocoa
  - Java
  - Win32

QuickTime in Modern Applications

- Some prime examples from Apple
  - iMovie
  - Final Cut Pro and Final Cut Express
  - iTunes
  - iPhoto
  - Keynote
  - Sherlock
  - iChat

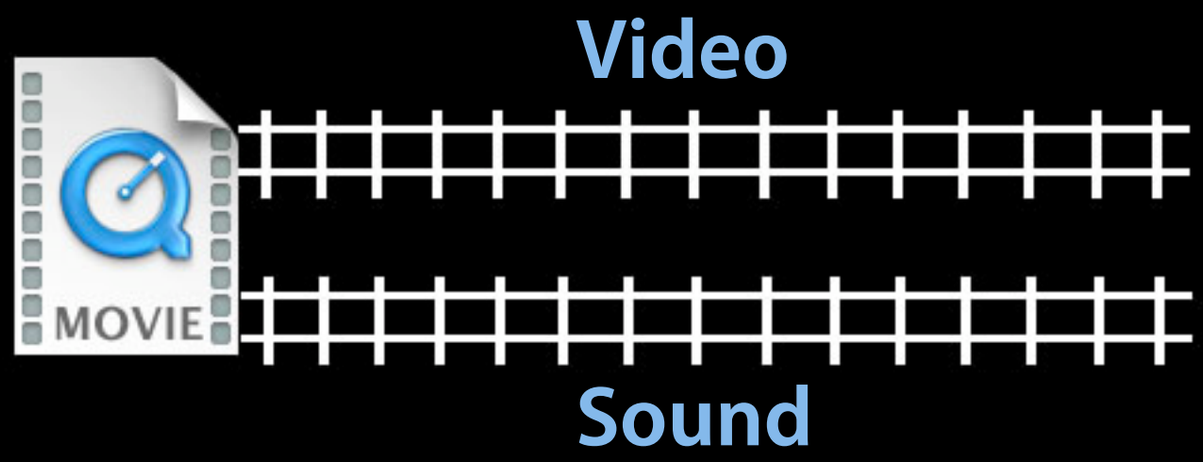
Demo

Digital Media the QT Way

- **Movie**
  - (moo-vee) n. A catalog of playable and editable content that describes the sources and types of content to be played together with its spatial and temporal organization.

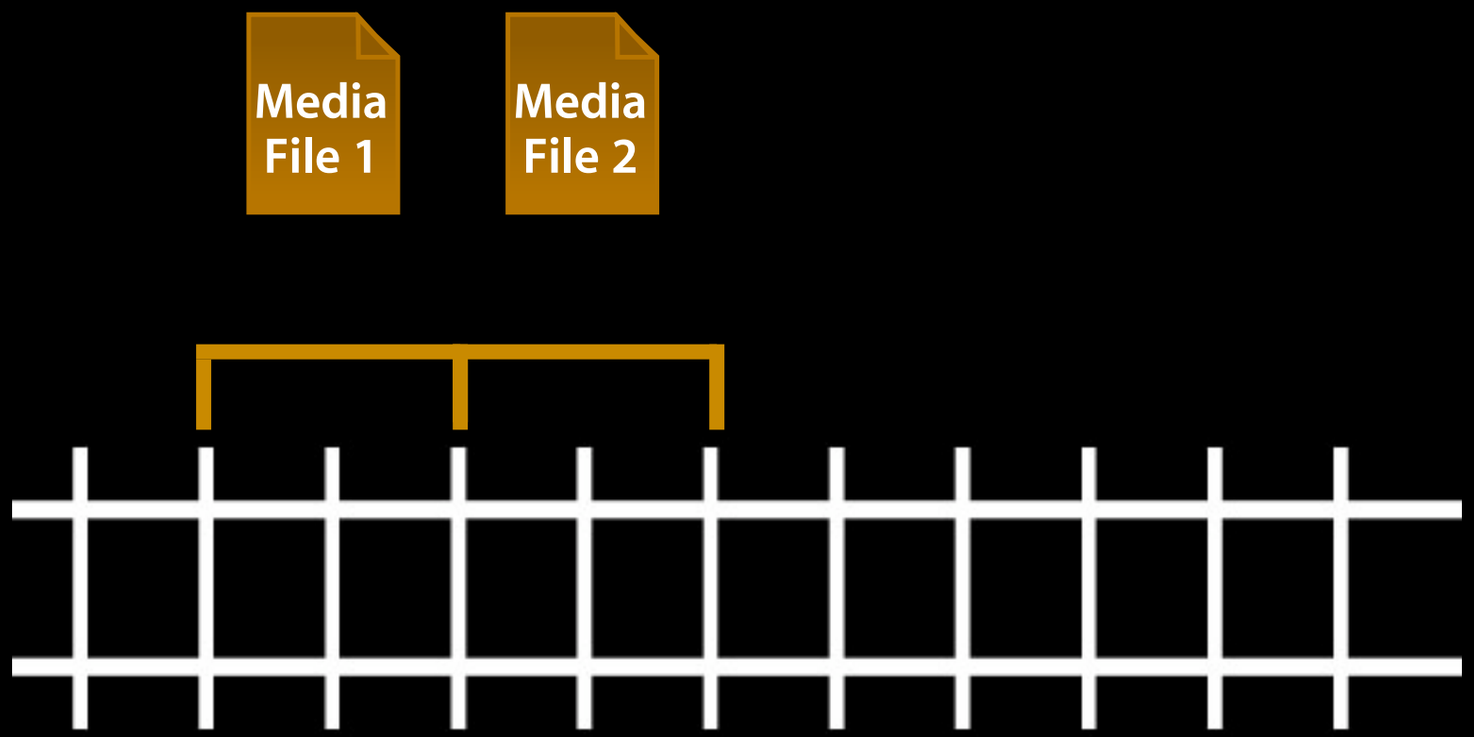
Digital Media the QT Way

- **Track**
  - (trak) n. A description of a single sequence of content with shared spatial and temporal properties and usually the same type and characteristics, e.g., a video track.



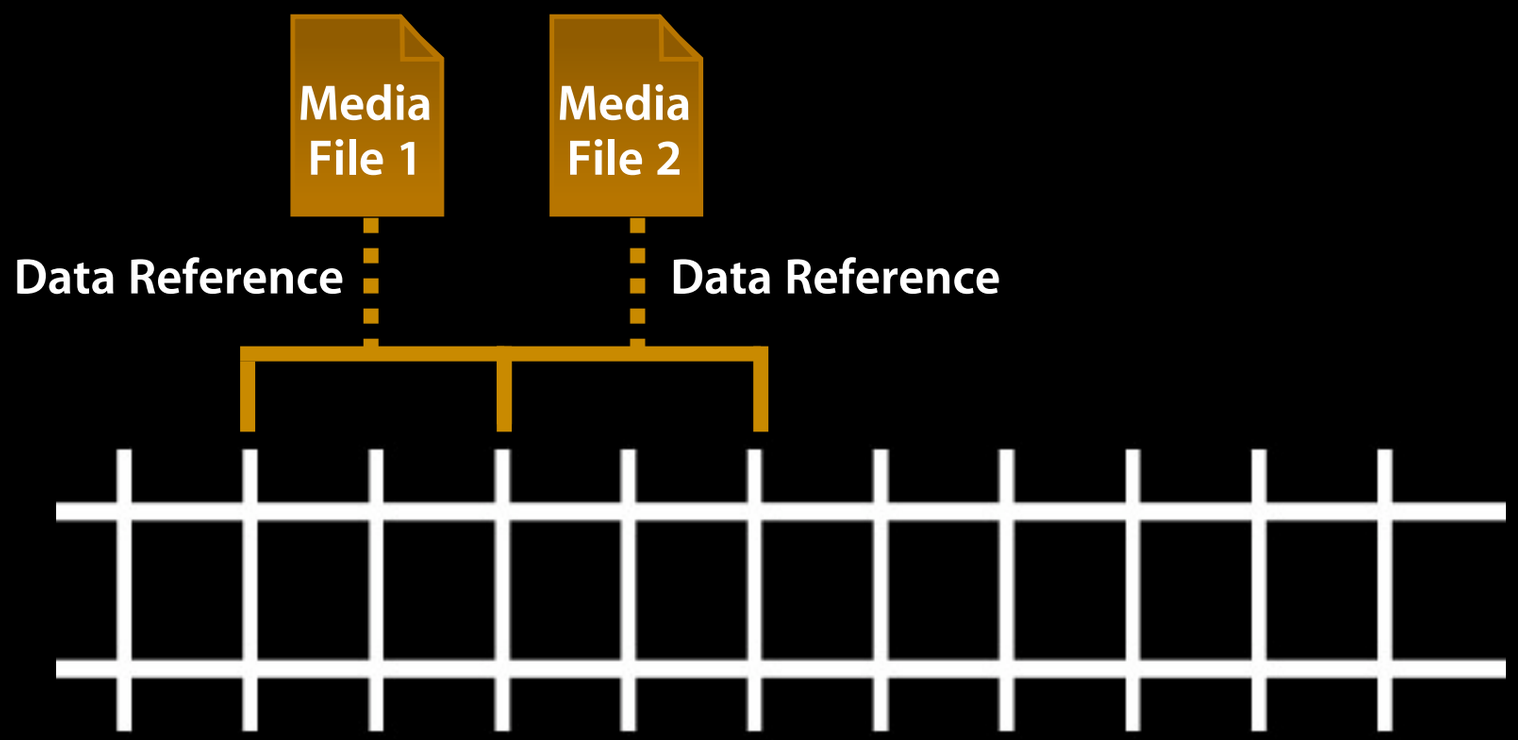
Digital Media the QT Way

- **Media**
  - (mee-dee-uh) n. (*singular*) A container of playable or editable content, e.g., an MPEG-4 file.



Digital Media the QT Way

- **Data Reference**
  - (day-tuh ref-er-uhnts) n. A semi-opaque handle-based data structure containing information required to locate media, e.g., a URL data reference.



Using These Simple Abstractions…

New Format Support Since Last Year

- MPEG-4
  - AAC
- JPEG 2000
- 3GPP
  - AMR
- WAV format 49 (MS-GSM)

What Has Changed Since 1990?

- File system
- Graphics subsystem
- Color matching
- User interface standards
- Text representation
- Audio subsystem
- Event handling
- Concurrency

Files and How to Specify Them

- Earth to QuickTime Team: What's an FSSpec?
- We have an abstraction: the data reference
- With data references, you can:
  - Read from files and other forms of storage
  - Write to files and other forms of storage
- Introducing the Data Ref Utilities
  - New APIs for creating data references

Data Reference Utilities

- A full set of APIs for creating QuickTime data references from various forms of input
- A sample:

```
OSErr QTNewDataReferenceFromFullPathCFString(
  CFStringRef    filePath,
  QTPathStyle    pathStyle,
  UInt32         flags,
  Handle *       outDataRef,
  OSType *       outDataRefType)
```

Data Reference Utilities

- See QuickTime/Movies.h for full set
- Highlights:
  - Files via FSRef
  - Files via FSSpec
  - Paths via CFStringRef
  - URLs via CFURLRef

First, 2 Words About EnterMovies()

- Call it
- Take care with calls upon the QuickTime framework from class methods, such as:

```
+ (void)initialize
{
  EnterMovies();
}

+ (BOOL)canOpenFile:(NSString *)path
{
  // wait for it, it’s coming up next...
}
```

Can QuickTime Open a File?

```
CFStringRef path;
Handle dataRef;
OSType dataRefType;
OSErr err;

err = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
err = CanQuickTimeOpenDataRef( dataRef, dataRefType,
   &canBeImage, &canBeMovie, &prefersToBeImage, 0 );
DisposeHandle(dataRef);
```

Movie From File

```
CFStringRef path;
Handle dataRef;
OSType dataRefType;
Movie qtMovie;
OSErr err;

if (!allowQTUserInteraction)
   flags |= newMovieDontAskUnresolvedDataRefs;
err = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
err = NewMovieFromDataRef( &qtMovie, flags,
      NULL, dataRef, dataRefType );
DisposeHandle(dataRef);
```

Movie From URL

```
NSURL *url;
Handle dataRef;
OSType dataRefType;
Movie qtMovie;
OSErr err;

if (!allowQTUserInteraction)
   flags |= newMovieDontAskUnresolvedDataRefs;
err = QTNewDataReferenceFromCFURL((CFURLRef)url,
   0, &dataRef, &dataRefType);
err = NewMovieFromDataRef( &qtMovie, flags,
      NULL, dataRef, dataRefType );
DisposeHandle(dataRef);
```

Write Movie to File

```
CFStringRef path;
Handle dataRef = NULL;
OSType dataRefType;
DataHandler dataHandler;

err = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
err = CreateMovieStorage(dataRef, dataRefType,
   kHFSCreatorCode, kScriptTag,
   createMovieFileDeleteCurFile, &dataHandler, NULL);
AddMovieToStorage(qtMovie, dataHandler);
CloseMovieStorage(dataHandler);
DisposeHandle(dataRef);
```

Write Movie to File (Cocoa Variant)

```
Movie qtMovie;
NSString *path;
Handle publicMovieHndl;
NSData *movieData;

publicMovieHndl = NewHandle(0);
PutMovieIntoHandle(qtMovie, publicMovieHndl);
HLock(publicMovieHndl);
movieData = [NSData dataWithBytes:*publicMovieHndl
   length:GetHandleSize(publicMovieHndl)];
DisposeHandle(publicMovieHndl);
[movieData writeToFile:path atomically:YES];

// set file attributes via NSFileManager
```

Flatten Movie to File

```
CFStringRef path;
Handle dataRef = NULL;
OSType dataRefType;
Movie newMovie;

err = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
newMovie = FlattenMovieDataToDataRef(qtMovie,
   flattenAddMovieToDataFork, dataRef, dataRefType,
   kHFSCreatorCode, smSystemScript,
   createMovieFileDeleteCurFile);
DisposeMovie(newMovie);
DisposeHandle(dataRef);
```

Examine Media Played by Movie

```
Movie qtMovie;
long  dataRefCount;
Handle dataRef, anchorDataRef;
OSType dataRefType, anchorDataRefType;
DataHandler dh;
Track track = GetMovieIndTrack(qtMovie, 1);
Media media = GetTrackMedia(track);

err = GetMovieAnchorDataRef(qtMovie, &anchorDataRef,
   &anchorDataRefType, NULL);
err = GetMediaDataRefCount(media, &dataRefCount);
err = GetMediaDataRef(media, 1, &dataRef,
   &dataRefType, NULL);
err = OpenADataHandler(dataRef, dataRefType,
   anchorDataRef, anchorDataRefType, NULL, 0, &dh);
```

Examine Media Played by Movie (Cont.)

```
// Having opened a data handler component for
// the media data ref, I can now fetch some info
// about the data ref.

// TO DO: *** I have got to stop talking to myself.

CFStringRef path;

err = QTGetDataHandlerFullPathCFString(dh,
   kQTNativeDefaultPathStyle, &path);

// clean up
CloseComponent(dh);
DisposeHandle(dataRef);
DisposeHandle(anchorDataRef);
```

Graphical Interlude

Unless otherwise noted...

Loading Media Data Into Memory

- Not a very good practice
- It is easy and it is fun, but it is just not hip

```
NSArray *paths = getMediaPaths(qtMovie);
NSEnumerator *enumerator = [paths objectEnumerator];
NSString *thisPath;

while (thisPath = (NSString *)[enumerator nextObject])
{
   NSData *mediaData = [NSData dataWithContentsOfFile:
      thisPath];

   // .. do something with media data
}
```

Export Sound to WAVE File

```
Handle dataRef = NULL;
OSType dataRefType;
DataHandler dataHandler;

err = QTNewDataReferenceFromFullPathCFString(
   (CFStringRef)path, kQTNativeDefaultPathStyle,
   0, &dataRef, &dataRefType);

SetMovieProgressProc(theMovie, (MovieProgressUPP)-1L, 0);
err = ConvertMovieToDataRef(theMovie, NULL, dataRef,
   dataRefType, kQTFileTypeWave, kHFSCreatorCode,
   0L, NULL);

DisposeHandle(dataRef);
```

Image From File

```
CFStringRef path;
Handle dataRef;
OSType dataRefType;
GraphicsImportComponent graphicsImporter;
ComponentResult result;

result = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
result = GetGraphicsImporterForDataRefWithFlags(dataRef,
   dataRefType, &graphicsImporter, 0);
DisposeHandle(dataRef);
```

Export Image to File

```
GraphicsImportComponent grip;
Handle dataRef = NULL;
OSType dataRefType;
GraphicsExportComponent graphicsExporter;
unsigned long sizeWritten;
ComponentResult result;

result = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
result = OpenADefaultComponent(
   GraphicsExporterComponentType, kQTFileTypePNG,
   &graphicsExporter);
```

Export Image to File (Cont.)

```
result = GraphicsExportSetInputGraphicsImporter(
   graphicsExporter, grip);
result = GraphicsExportSetOutputDataReference(
   graphicsExporter, dataRef, dataRefType);
result = GraphicsExportDoExport(graphicsExporter,
   &sizeWritten);

CloseComponent(graphicsExporter);
DisposeHandle(dataRef);
```

Graphics

- Earth to QuickTime: what's a GWorld?
- New graphics subsystems
- We have intermediate APIs using QuickDraw
- Movies and Still Images
  - Rendering to destinations various
- Still images and Quartz
  - GraphicsImportCreateCGImage
  - GraphicsExportSetInputCGImage

Play Movie in Window

```
WindowRef window;
Movie     qtMovie;
Rect      qdRect;
CGrafPtr  windowPort;
OSErr     err;

SetMovieBox(qtMovie, movieBounds);

// window = [myNSWindow windowRef];
windowPort = GetWindowPort(window);
SetMovieGWorld(qtMovie, windowPort, NULL);

// set the movie's rate to start it playing
```

Play Movie Offscreen

```
size_t    bytesPerRow;
void     *buffer;
CGSize    outputSize;
unsigned long pixelFormat;
Rect      qdRect;
GWorldPtr gWorld = NULL;
OSErr     err;

SetRect(&qdRect, 0, 0, outputSize.width,
   outputSize.height);
err = NewGWorldFromPtr(&gWorld, pixelFormat,
   &qdRect, NULL, NULL, 0, buffer, bytesPerRow);
SetMovieGWorld(qtMovie, gWorld, NULL);
```

Draw Image Offscreen

```
GraphicsImportComponent grip;
size_t    bytesPerRow;
void     *buffer;
CGSize    outputSize;
unsigned long pixelFormat;
Rect      qdRect;
GWorldPtr gWorld = NULL;
OSErr     err;

SetRect(&qdRect, 0, 0, outputSize.width,
   outputSize.height);
err = NewGWorldFromPtr(&gWorld, pixelFormat,
   &qdRect, NULL, NULL, 0, buffer, bytesPerRow);
GraphicsImportSetGWorld(grip, gWorld, NULL);
GraphicsImportDraw(grip);
```

Draw Movie Frame to CGContext

```
TimeValue movieTime;
PicHandle picHndl;
CGDataProviderRef dataProvider;
QDPictRef pictDataRef;
OSStatus result;

movieTime = GetMovieTime(qtMovie, NULL);
picHndl = GetMoviePict(qtMovie, movieTime);
HLock((Handle)picHndl);
dataProvider = CGDataProviderCreateWithData ( NULL,
   *picHndl, GetHandleSize((Handle)picHndl), NULL );
pictDataRef = QDPictCreateWithProvider(dataProvider);
result = QDPictDrawToCGContext(ctx, drawRect,
   pictDataRef);
```

Draw Movie Frame to CGContext

```
// don’t forget to clean up!

QDPictRelease(pictDataRef);
CGDataProviderRelease(dataProvider);
KillPicture(picHndl);
```

CGImage From File

```
CFStringRef path;
CGImageRef imageRef = NULL;
Handle dataRef = NULL;
OSType dataRefType;
GraphicsImportComponent gi;
ComponentResult result;

result = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
result = GetGraphicsImporterForDataRefWithFlags(dataRef,
   dataRefType, &gi, 0);
result = GraphicsImportCreateCGImage(gi, &imageRef, 0);
DisposeHandle(dataRef);
CloseComponent(gi);

return CGImageRef;
```

CGImage to File

```
CFStringRef path;
CGImageRef imageRef;
Handle dataRef = NULL;
OSType dataRefType;
GraphicsExportComponent graphicsExporter;
unsigned long sizeWritten;
ComponentResult result;

result = QTNewDataReferenceFromFullPathCFString(path,
   kQTNativeDefaultPathStyle, 0, &dataRef, &dataRefType);
result = OpenADefaultComponent(
   GraphicsExporterComponentType, kQTFileTypePNG,
   &graphicsExporter);
result = GraphicsExportSetInputCGImage(
   graphicsExporter, imageRef);
```

CGImage to File (Cont.)

```
result = GraphicsExportSetOutputDataReference(
   graphicsExporter, dataRef, dataRefType);
result = GraphicsExportDoExport(graphicsExporter,
   &sizeWritten);

CloseComponent(graphicsExporter);
DisposeHandle(dataRef);
```

Color Matching

- Earth to QuickTime: Why aren't my images color-matched?
- Image files can contain embedded ColorSync profiles
- QuickTime provides services to accessing these profiles
- Now, Graphics Importers call ColorSync to match while drawing
- And they attempt to preserve profiles when converting images from one format to another
- More details: Session 716 Advanced QT Programming Techniques

Text and How to Encode It

- Earth to QuickTime team: what's a script tag?
- Better support for Unicode in text media
  - On Mac OS X, ATSUI instead of QuickDraw text
  - More faithful rendering
  - Loss of Shadow and Outline styles
- Vertical text and alignment for 3GPP
- Text search APIs take string to match as Unicode

Demo

New Horizons in UI Design

- Earth to QuickTime: what is a modal dialog?
- We do sheets
  - Standard compression
  - Effects
- We factor
  - Component properties

Standard Audio Compression Sheet

```
NSWindow *parentWindow;
ComponentInstance ci;
Handle settingsHndl = NULL;
ComponentResult result;
SCWindowSettings windowSettings;
NSData *settingsAsData = nil;

ci = OpenDefaultComponent(StandardCompressionType,
   StandardCompressionSubTypeSound);

windowSettings.size = sizeof(SCWindowSettings);
windowSettings.windowRefKind = scWindowRefKindCarbon;
windowSettings.parentWindow = [parentWindow windowRef];
(void)SCSetInfo(ci, scWindowOptionsType, &windowSettings);
```

Standard Audio Compression Sheet (Cont.)

```
// Get compression settings from the user.
result = SCRequestImageSettings(ci);

// Get the compression settings
result = SCGetInfo(ci, scSettingsStateType, &settingsHndl);

HLock(settingsHndl);
settingsAsData = [NSData dataWithBytes:*settingsHndl
   length:GetHandleSize(settingsHndl)];
DisposeHandle(settingsHndl);

CloseComponent(ci);
```

Effects Parameter Sheet

```
QTAtomContainer      fxList = NULL;
long                 minSrcs = 2;
long                 maxSrcs = 2;
QTEffectListOptions  lOpts = 0;
QTEffectListOptions  dOpts = pdOptionsDisplayAsSheet;
QTParameterDialog    paramDlg;
QTAtomContainer      fxDesc = nil;
OSErr                myErr;

myErr = QTGetEffectsList ( &fxList, minSrcs,
   maxSrcs, lOpts );
myErr = QTNewAtomContainer ( &fxDesc );
myErr = QTCreateStandardParameterDialog ( fxList,
   fxDesc, dOpts, &paramDlg );
```

Effects Parameter Sheet (Cont.)

```
QTEventLoopDescriptionRecord eld;

eld.recordSize = sizeof(eld);
eld.windowRefKind = kEffectParentWindowCarbon;
eld.parentWindow = [parentWindow windowRef];
eld.eventTarget = NULL;

QTStandardParameterDialogDoAction(paramDlg,
   pdActionRunInEventLoop, &eld);

QTDismissStandardParameterDialog(gParamDlg);
```

Demo

Component Properties

- A standard means to factor settings from UI
- Intended for use by standard compression, exporters, importers, and other types of components

```
ComponentResult QTSetComponentProperty(
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress);
```

Threading

- Earth to QuickTime: why can't I call your APIs from an arbitrary thread?
- Now, some you can, with great care, on Panther only
- Graphics import and export with some formats
- Video processing with some video codecs
- Audio processing to follow
- More details: Session 716 Advanced QT Programming Techniques

Events and How to Process Them

- Earth to QuickTime team: what is WaitNextEvent?
- We offer higher-level modules
  - Carbon movie control
  - Cocoa class: NSMovieView
  - ActiveX control

Audio

- Earth to QuickTime team: ever hear of surround sound?
- More than two discrete channels
- Use of Core Audio in QuickTime
- Using Core Audio with QuickTime
  - Audio extraction
- More details: Session 711—Audio and QuickTime

Answers to Common Questions

- How do I open a movie, given:
  - A CFString?
  - A CFURL?
- How do I specify where the movie should render, given:
  - A CGContext?
- Where and when should I call MCIdle or MoviesTask?
- How do I convert an NSTimeInterval to a TimeValue?
- Not a common question? Come to the lab

Tasking Your Movies

```
long delay;
CFTimeInterval nextFireDate;

if (CFSetGetCount(_primaryObjects))
  CFSetApplyFunction(_primaryObjects,
    TaskFunction, NULL);
else
    MoviesTask(DoTheRightThing, DoTheRightThing);

QTGetTimeUntilNextTask(&delay, 1000);
if (delay < kMinimumIdleIntervalInSeconds)
  nextFireDate = kMinimumIdleIntervalInSeconds;
else
  nextFireDate = CFAbsoluteTimeGetCurrent() +
    (CFTimeInterval)(delay/1000.0);
CFRunLoopTimerSetNextFireDate(timer, nextFireDate);
```

Tasking Your Movies (Cont.)

```
void TaskFunction(const void *value, void *context)
{
  // imagining that the value is the id of an NSObject
  // to be tasked

  // and further imagining that “idle” is a message
  // handled by the object by calling
  // MCIdle or MoviesTask
  [(id)value idle];
}
```

NSTimeInterval to TimeRecord

```
NSTimeInterval timeInterval;
double fraction;
int exponent;
TimeScale bestScale;

fraction = frexp(timeInterval, &exponent);
if (exponent <= 44)
   bestScale = 1000000;
else if (exponent <= 47)
   bestScale = 90000;
else if (exponent <= 54)
   bestScale = 600;
else if (exponent < 64)
   bestScale = 1;
else
   return FALSE;
```

NSTimeInterval to TimeRecord (Cont.)

```
TimeRecord tr;

// special case for 0
if (0.0 == fraction && 0 == exponent)
{
   tr.value.hi = tr.value.lo = 0;
}
else
{
   *(long long *)&tr.value =
      (long long)(bestScale * timeInterval);
}

tr.scale = bestScale;
tr.base = NULL;
return TRUE;
```

You Can Use Higher-Level Objects

- Carbon Movie control
- Cocoa NSMovieView
- Java Component and JComponent

Carbon Movie Control

- Works with Carbon Control Manager
- QuickTime 6.4 revision will work with Carbon HI Views
- How to create one:

```
OSErr CreateMovieControl (
  WindowRef      theWindow,
  Rect           *localRect,
  Movie          theMovie,
  UInt32         options,
  ControlRef     *returnedControl );
```

Demo

QuickTime for Java 6.1

- For JDK 1.3 and JDK 1.4.1 on Windows and on Mac OS X
  - Complete, up-to-date set of bindings to QuickTime APIs
  - New quicktime.app.view package
    - Java AWT Component
    - Java Swing JComponent
  - Some code migration may be necessary

quicktime.app.view Package

- QTFactory
- QTImageProducer
- MoviePlayer
- GraphicsImporterDrawer
- QTComponent
- QTJComponent

Movie From File, AWT Component

```
//create movie from QTFile
QTFile qtfile = new QTFile (moviePath);
Movie mov = Movie.fromFile(qtfile);
MovieController mc = new MovieController(mov);

//create component using factory methods
QTComponent movComp = QTFactory.fromMovieController(mc);
frame.add((Component)movComp);
frame.show();
```

Movie From File, JComponent

```
//create movie from QTFile
QTFile qtfile = new QTFile (moviePath);
Movie mov = Movie.fromFile(qtfile);
MoviePlayer movPlayer = new MoviePlayer(mov);

//create component using factory methods
QTJComponent  movComponent =
  QTFactory.fromMoviePlayer(movPlayer);
jframe.getConentPane().add((JComponent)movComponent);

jframe.show();
```

Deprecated Packages

- quicktime.app.actions
- quicktime.app.anim
- quicktime.app.audio
- quicktime.app.display
- quicktime.app.event
- quicktime.app.image
- quicktime.app.players
- quicktime.app.sg
- quicktime.app.spaces
- quicktime.app.ui

Demo

AppKit Movie Support

- NSMovie and NSMovieView
  - Mature fundamental support in Panther
- An example of usage:

```
NSMovieView *movieView;
NSURL *url;

NSMovie *movie = [[[NSMovie alloc]
  initWithURL:url byReference:NO] autorelease];
[movieView setMovie:movie];
```

QuickTime for Cocoa

- QTMovie
- QTMovieView
- QTTrack
- QTTrackEnumerator
- QTMedia
- QTDataReference
- QTDataReferenceEnumerator
- QTTime

QT for Cocoa

- Seeding this summer
- Contact: qtcocoaseed@apple.com, with:
  - Name
  - Company
  - Product
  - Goals for participating in seed
  - Describe value of opportunities

Demo

What About More Elaborate Needs?

- In each of these environments, you have access to QuickTime C APIs
- We update and expand our API set as requirements demand
- This year's hot topics
  - Threading
  - Audio
  - ColorSync

Wrap Up

Roadmap

701 QuickTime State of the Union	The Presidio Tues., 9:00am
711 Audio and QuickTime	Russian Hill Wed., 9:00am
716 Advanced QuickTime Programming Techniques	Pacific Heights Wed., 2:00pm
727 QuickTime Streaming Server Programming	The Mission Thurs., 2:00pm
730 QuickTime Alternative Programming Environments	Pacific Heights Fri., 10:30pm
FF011 QuickTime Feedback Forum	Haight/Ashbury Thurs., 3:30pm

Who to Contact

Guillermo Ortiz
QuickTime Technology Manager
quicktimeman@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

QuickTime Overview
ADC Home > Documentation > QuickTime >

What's New in QuickTime 6.1
ADC Home > Documentation > QuickTime >

QuickTime API Reference
ADC Home > Documentation > QuickTime >

QuickTime for Java
ADC Home > QuickTime > QTJava >

For More Information

Documentation (Cont.)

General Information
http://www.apple.com/quicktime

Developer Information
http://developer.apple.com/documentation

QuickTime Information
http://developer.apple.com/QuickTime

Do Not Miss!

Lab: QuickTime Content Creation

Meet with QuickTime content experts for daily content specific topics that are of special interest to you. Bring your own media and PowerBook. No appointment is needed—first-come, first-served.

Sacramento Room

Lab: QuickTime Development

Meet with Apple's QuickTime engineers. Each day will emphasize a specific QuickTime development topic. Bring your own code and PowerBook. No appointment is needed—first-come, first-served.

Sacramento Room

Lab Hours

Monday	12:00pm to 6:00pm
Tuesday	9:00am to 6:00pm
Wednesday	9:00am to 6:00pm
Thursday	9:00am to 4:30pm
Friday	9:00am to 6:00pm

Q&A

Kevin Calhoun
Software Engineer
kcalhoun@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference